CS267
Applications of Parallel Computers

Lecture 1: Introduction

David H. Bailey

Based on previous notes by

Prof. Jim Demmel and Prof. David Culler

dhbailey@lbl.gov

http://www.nersc.gov/~dhbailey/cs267

Outline

- Introductions
- Why large important problems require the capabilities of powerful computers
- Why powerful computers must be parallel processors
- Structure of the course

Administrative Information

- **Instructors:**
  - David H. Bailey, LBL 50B-2239, dhbailey@lbl.gov
  - Robert F. Lucas, LBL 50B-2245, rflucas@lbl.gov
  - TA: Edward Jason Reidy, xxx Soda, ejr@cs.berkeley.edu
- **Office hours (Soda office is being arranged)**
  - T Th 12:30pm to 1:30pm, and by appointment
- **Accounts and others -- fill out online registration!**
- **Class survey -- fill out online!**
- **Discussion section: TBD, based on survey**
- **Most class material and lecture notes are at: www.nersc.gov/~dhbailey/cs267**

Why we need powerful computers

Units of Measurement in High Performance Computing

- Mflop/s — 10^6 flop/sec
- Gflop/s — 10^9 flop/sec
- Tflop/s — 10^{12} flop/sec
- Pflop/s — 10^{15} flop/sec
- Mbyte — 10^6 byte (also 2^{20} = 1048576)
- Gbyte — 10^9 byte (also 2^{30} = 1073741824)
- Tbyte — 10^{12} byte (also 2^{40} = 10995211627776)
- Pbyte — 10^{15} byte (also 2^{50} = 1125899906842624)

Why we need powerful computers

- **Traditional scientific and engineering paradigm:**
  - **Do theory or paper design.**
  - **Perform experiments or build system.**
- **Limitations:**
  - **Too difficult -- build large wind tunnels.**
  - **Too expensive -- build a throw-away passenger jet.**
  - **Too slow -- wait for climate or galactic evolution.**
  - **Too dangerous -- weapons, drug design, climate experimentation.**
- **Computational science paradigm:**
  - **Use high performance computer systems to model the phenomenon in detail, using known physical laws and efficient numerical methods.**

The economic impact of high performance computing

- **Airlines:**
  - Large airlines recently implemented system-wide logistic optimization systems on parallel computer systems.
  - Savings: approx. $100 million per airline per year.
- **Automotive design:**
  - Major automotive companies use large systems for CAD-CAM, crash testing, structural integrity and aerodynamic simulation. One company has 500+ CPU parallel system.
  - Savings: approx. $1 billion per company per year.
- **Semiconductor industry:**
  - Large semiconductor firms have recently acquired very large parallel computer systems (500+ CPUs) for device electronics simulation and logic validation (ie prevent Pentium divide fiasco).
  - Savings: approx. $1 billion per company per year.
- **Securities industry:**
  - Savings: approx. $15 billion per year for U.S. home mortgages.

Some Particularly Challenging Computations

- Global climate modeling
- Crash simulation
- Astrophysical modeling
- Earthquake and structural modeling
- Medical studies -- i.e. genome data analysis
- Phylogeny -- evolutionary history of species
- Web service and web search engines
- Financial and economic modeling
- Transaction processing
- Drug design -- i.e. protein folding
- Nuclear weapons -- test by simulations

Global Climate Modeling

- Function of four arguments: longitude, lattitude, elevation, time; which returns six values: temp, press, humidity, wind velocity.
- To model this on a computer we:
  - Discretize the domain using a finite grid, e.g., points 1 km apart.
  - Devise an algorithm to predict weather at time t+1 from time t.
  - Solve Navier-Stokes equations for fluid flow of atmosphere -- roughly 100 flops per grid point with a 1 min time step.
  - To match real time we need 5×10^{11} flops in 60 sec = 8 Gflop/s.
  - Weather prediction (7 days in 24 hours) => 56 Gflop/s.
  - Climate prediction (50 years in 30 days) => 4.8 Tflop/s.
  - To use in policy negotiations (12 hours) => 288 Tflop/s.
  - For a grid with twice the resolution in each dimension, multiply the above figures by at least eight.
- Current models use much coarser: www-fp.mcs.anl.gov/chammp

Heart Simulation

- Many biological structures can be modeled as an elastic structure in an incompressible fluid.
- Using the “immersed boundary method” involves solving Navier-Stokes equations, plus some feature-specific computations on the various organ components [Peskin&McQueen].
- 20 years of development in model, used to design artificial valves.
- 64^3 was possible on Cray YMP, but 128^3 required for accurate model (would have taken 3 years).
- Done on a Cray C90 -- could use 100x faster and 100x more memory.
- More computing power would yield a more accurate model, and ultimately one that could be used in real-time clinical work.

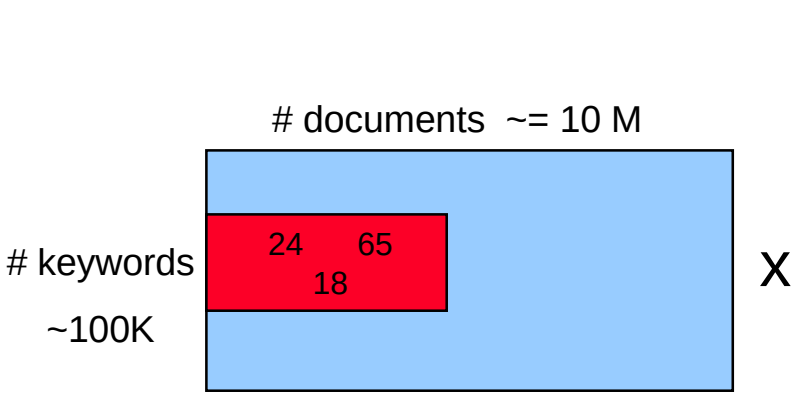
Parallel Computing in Web Search

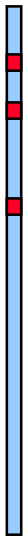
- Functional parallelism: crawling, indexing, sorting
- Parallelism between queries: multiple users
- Finding information amidst junk
- Preprocessing of the web data set to help find information

- General themes of sifting through large, unstructured data sets:
  - when to put white socks on sale
  - what kind of junk mail should you receive
  - finding medical problems in a community

Application: Finding Useful Documents on Web

° **One algorithm, Latent Semantic Indexing (LSI), needs large sparse matrix-vector multiply**





- Matrix is compressed
- "Random" memory access
- Scatter/gather vs. cache miss per 2Flops

° **Ten million documents in typical matrix.**

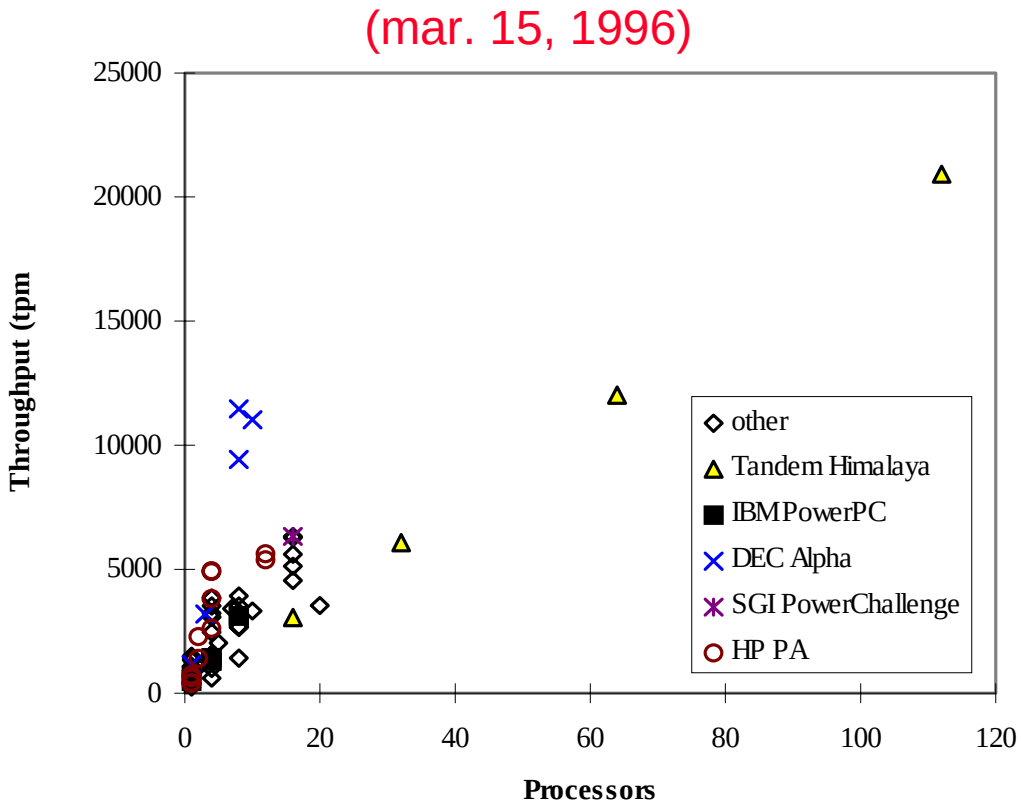
° **Web storage increasing 2x every 5 months.**

° **Similar ideas may apply to image retrieval.**

Latent Semantic Indexing (LSI) Challenges

- On conventional microprocessor node:
  - UltraSparc 166 MHz, 330 Mflop/s peak, Cache miss is 300 ns.
  - Matrix-vector multiply, does roughly 3 loads and 2 flops, with 1.37 cache misses on average.
  - ~4.5 Mflop/s (2-5 Mflop/s measured).
  - Memory accesses are irregular.
- On Cray T3E:
  - Osni Marques of LBL parallelized code for the T3E.
  - Performance scales nearly linearly with number of nodes used.
- Implementation is also I/O intensive.

Transaction Processing

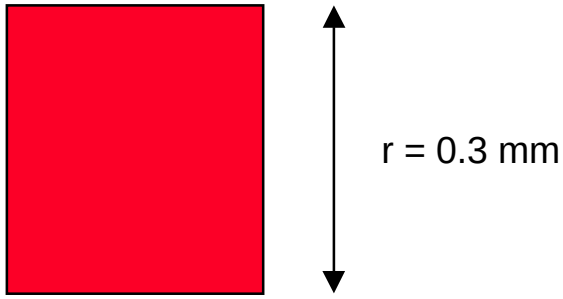


- **Parallelism is natural in relational operators: select, join, etc.**
- **Many difficult issues: data partitioning, locking, threading.**

Why powerful computers are parallel

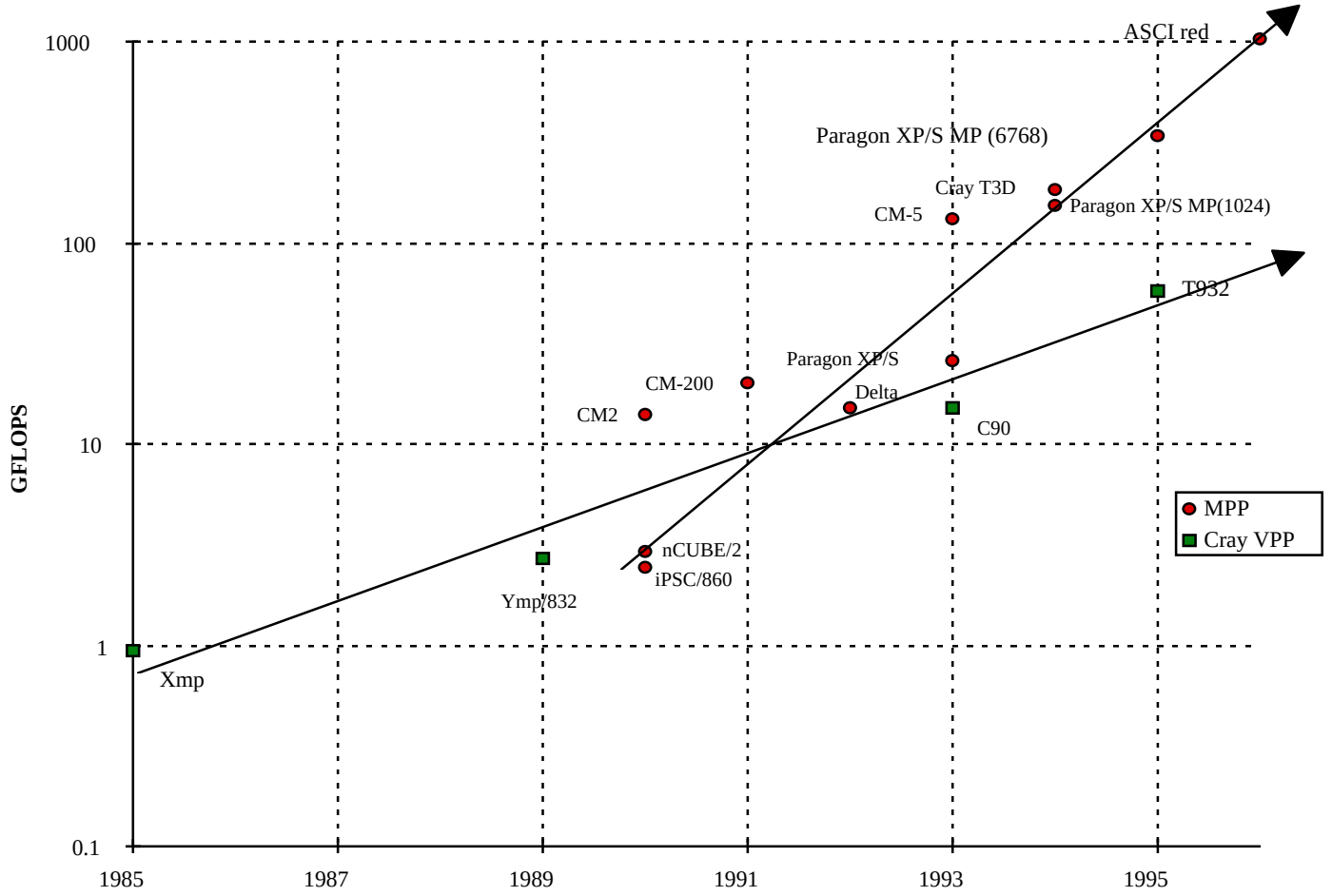
How fast can a serial computer be?

1 Tflop/s, 1 Tbyte sequential machine



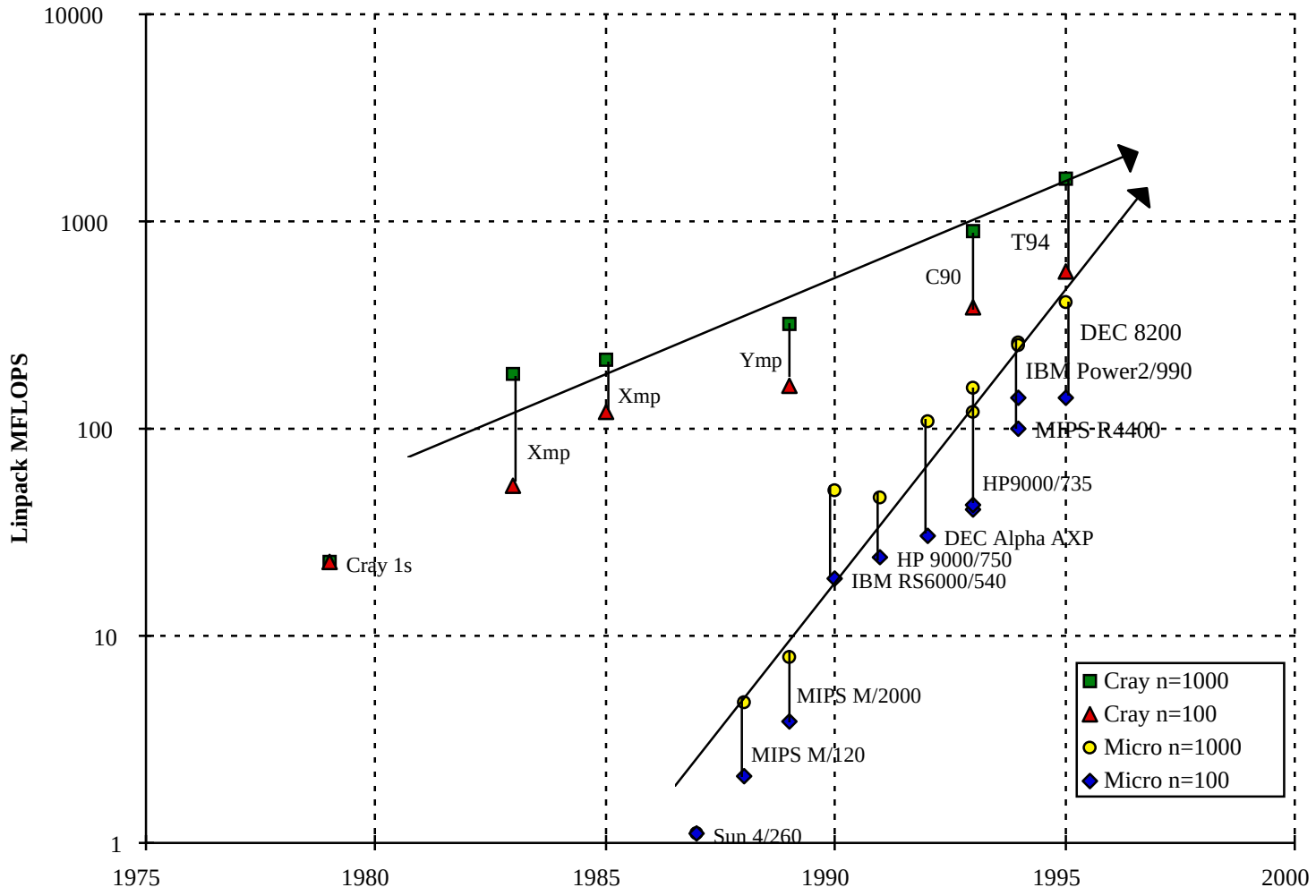
- **Consider the 1 Tflop/s sequential machine:**
  - **Data must travel some distance, r, to get from memory to CPU.**
  - **Go get 1 data element per cycle, this means 10^{12} times per second at the speed of light, c = $3x10^8$ m/s. Thus r < $c/10^{12}$ = 0.3 mm.**
- **Now put 1 Tbyte of storage in a 0.3 mm x 0.3 mm area:**
  - **Each word occupies about 3 square Angstroms, or the size of a small atom.**

Trends in Parallel Computing Performance

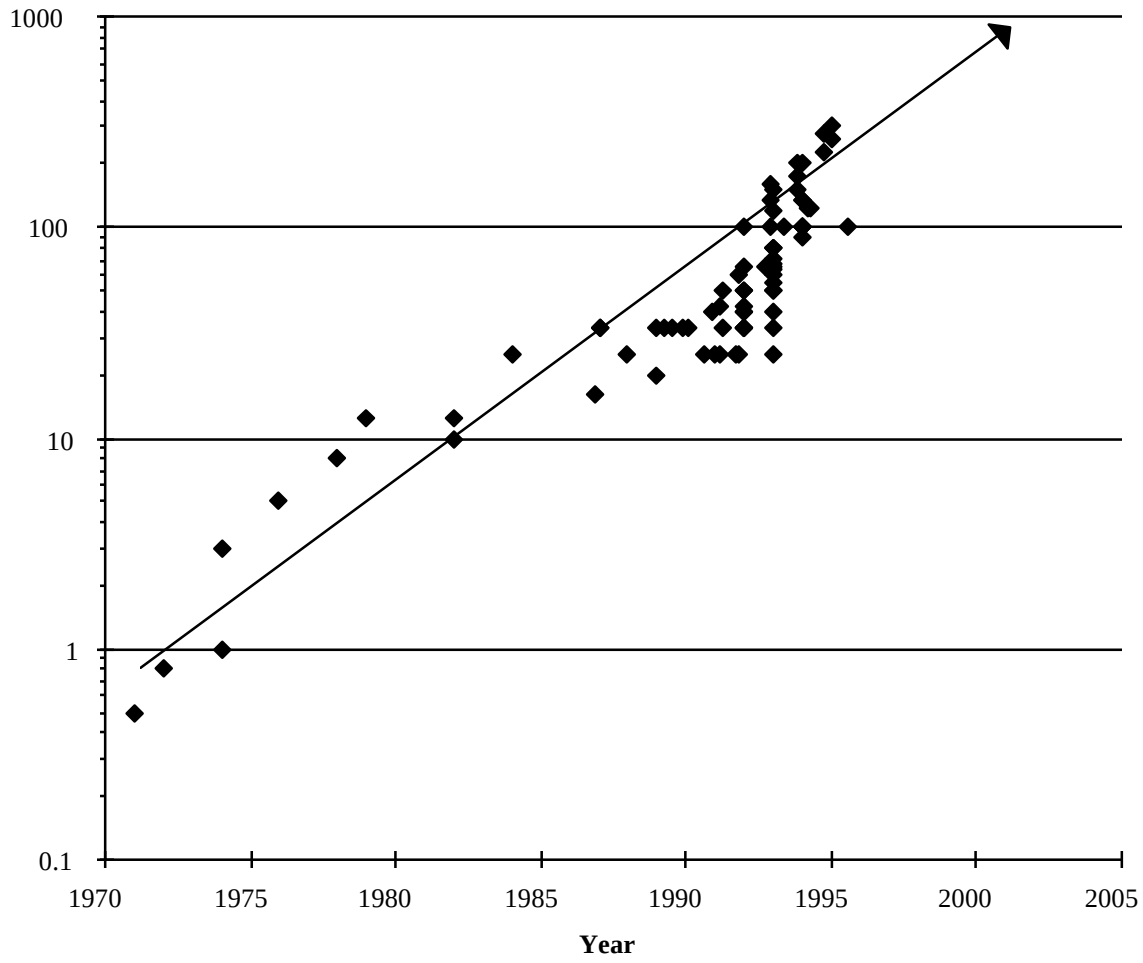


- 1 Tflop/s on Linpack, 12/16/96, ASCI Red (7264 Intel processors)
- Up to 1.6 Tflop/s by 1/99, on ASCI Blue (5040 SGI R10ks)
- See performance.netlib.org/performance/html/PDStop.html

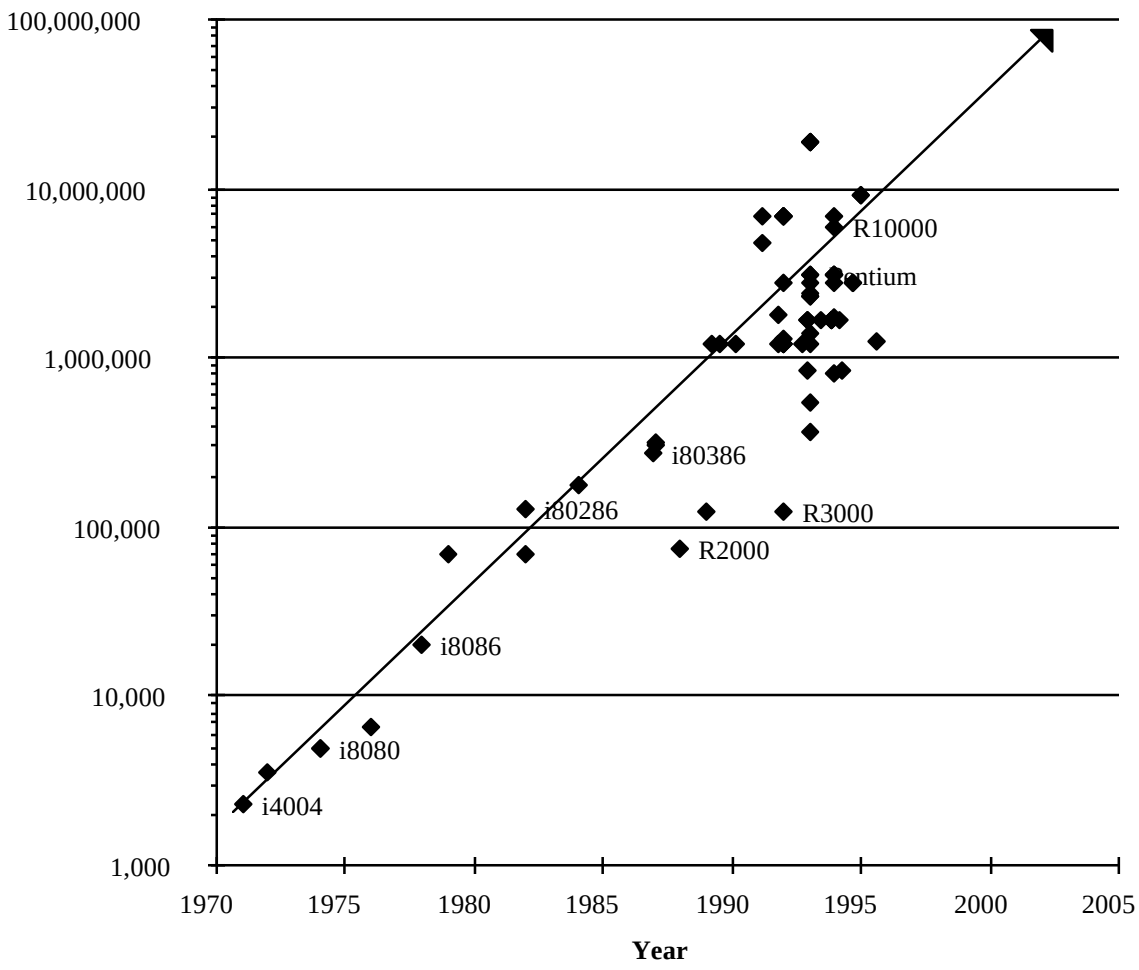
Empirical Trends: Microprocessor Performance



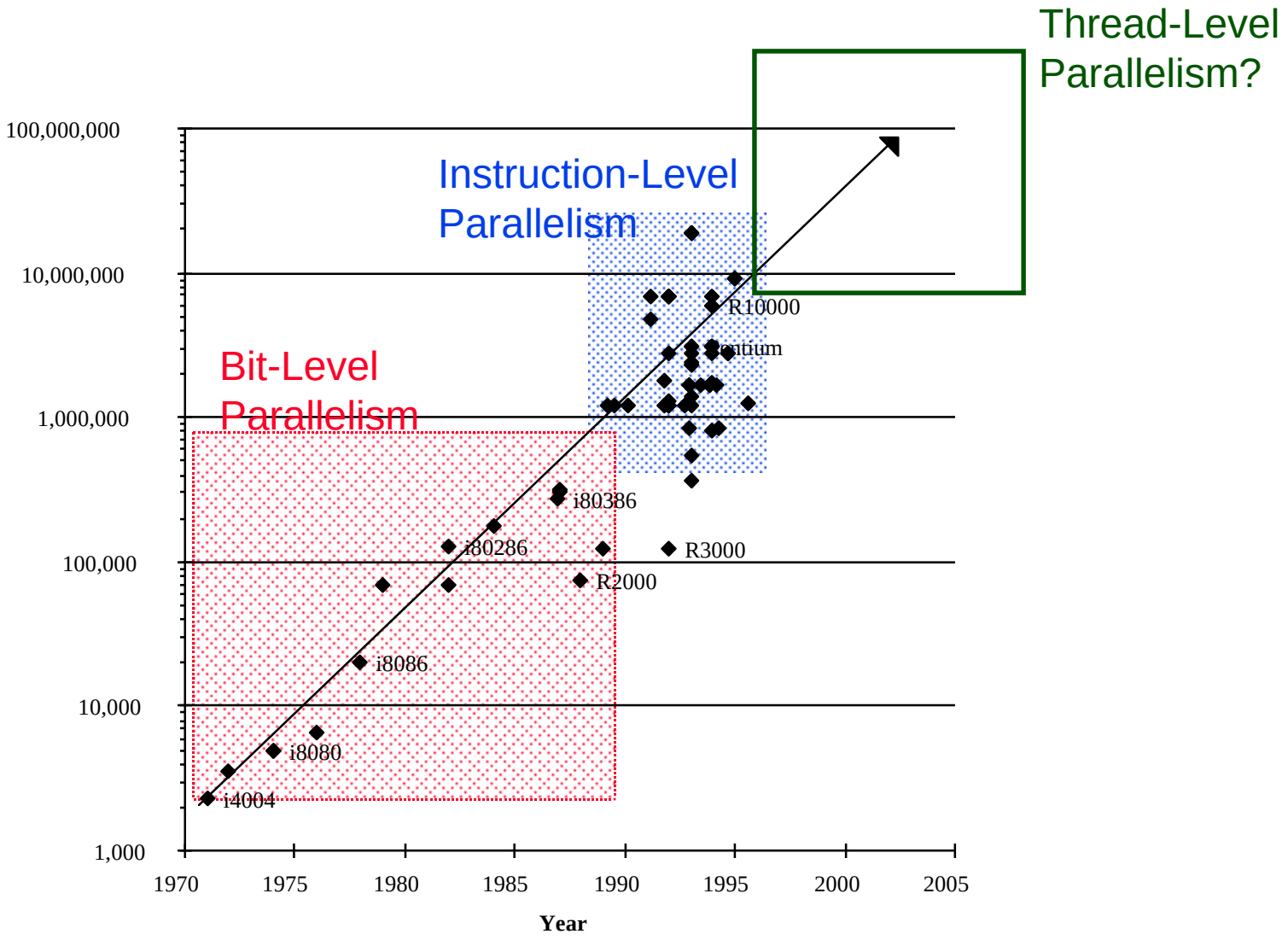
Microprocessor Clock Rate



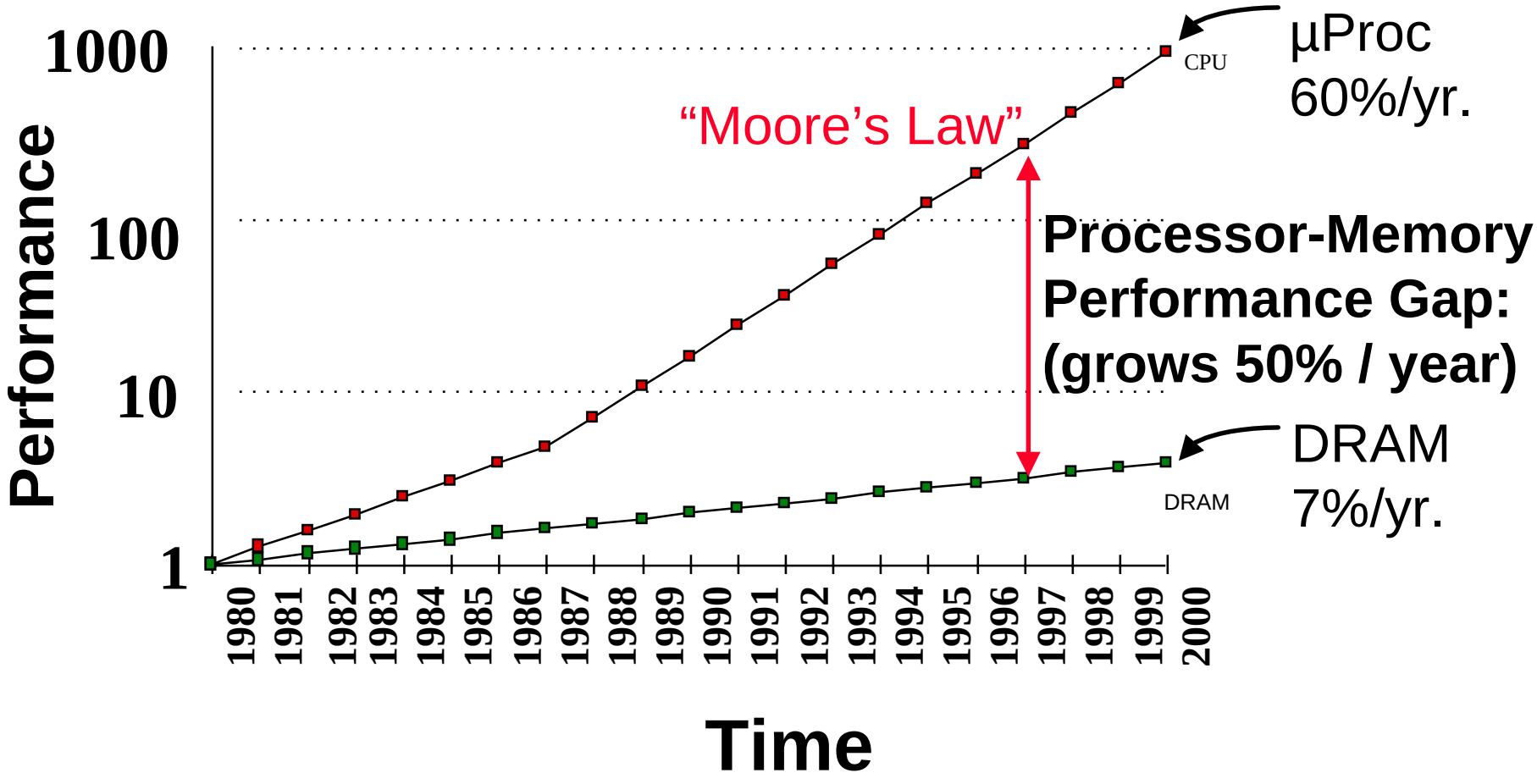
Microprocessor Transistors



Microprocessor Transistors and Parallelism



Processor-DRAM Gap (latency)



Impact of Device Shrinkage

- What happens when the feature size shrinks by a factor of x ?
- Clock rate goes up by x
  - actually less than x, because of power consumption
- Transistors per unit area goes up by x^2
- Die size also tends to increase
  - typically another factor of ~x
- Raw computing power of the chip goes up by ~ x^4 !
  - of which x^3 is devoted either to parallelism or locality

Principles of Parallel Computing

- Parallelism and Amdahl's Law
- Finding and exploiting granularity
- Preserving data locality
- Load balancing
- Coordination and synchronization
- Performance modeling

All of these things make parallel programming more difficult than sequential programming.

“Automatic” Parallelism in Modern Machines

- Bit level parallelism: within floating point operations, etc.
- Instruction level parallelism (ILP): multiple instructions execute per clock cycle.
- Memory system parallelism: overlap of memory operations with computation.
- OS parallelism: multiple jobs run in parallel on commodity SMPs.

There are limitations to all of these!

Thus to achieve high performance, the programmer needs to identify, schedule and coordinate parallel tasks and data.

Finding Enough Parallelism

- **Suppose only part of an application seems parallel**
- **Amdahl's law**
  - **Let s be the fraction of work done sequentially, so (1-s) is fraction parallelizable.**
  - **P = number of processors.**

$$\text{Speedup}(P) = \text{Time}(1)/\text{Time}(P)$$

$$<= 1/(s + (1-s)/P)$$

$$<= 1/s$$

- **Even if the parallel part speeds up perfectly, we may be limited by the sequential portion of code.**

Little's Law

Concurrency = latency x bandwidth

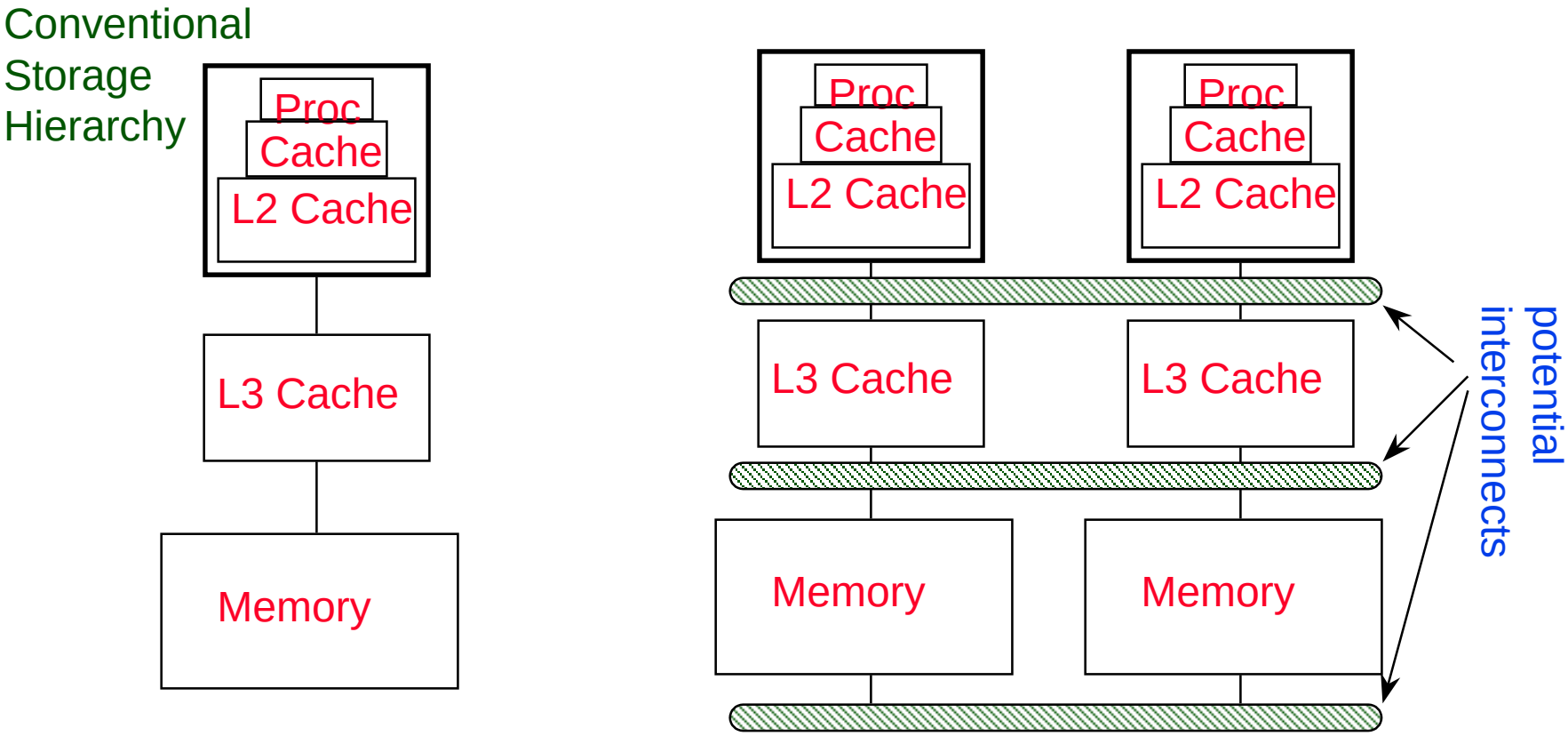
Example:

- **1000 processor system, 1 GHz clock, 100 ns memory latency, and maintains 100 words of memory in data paths between CPU and memory.**
- **Note main memory bandwidth ~ 1000 x 100 words x 10^9/s = 10^{14} words/sec.**
- **Then an application must have roughly 10^{-7} x 10^{14} = 10^7 way concurrency to achieve full performance potential of system.**

Overhead of Parallelism

- Given enough parallel work, this is the most significant barrier to getting desired speedup.
- Parallelism overheads include:
  - cost of starting a thread or process
  - cost of communicating shared data
  - cost of synchronizing
  - extra (redundant) computation
- Each of these can be in the range of milliseconds (= millions of flops) on some systems
- Tradeoff: Algorithm needs sufficiently large units of work to run fast in parallel (i.e. large granularity), but not so large that there is not enough parallel work.

Locality and Parallelism



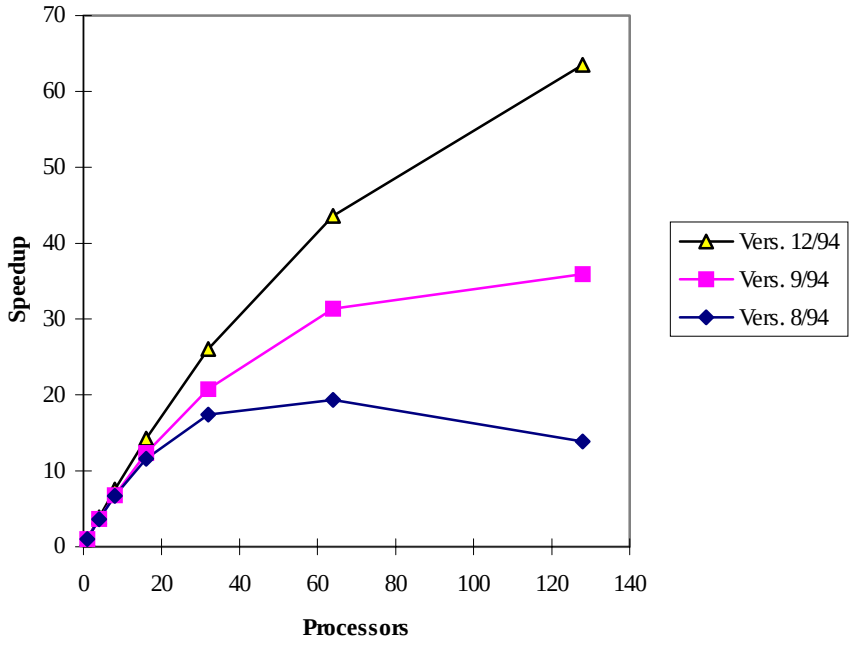
- **Large memories are slow, fast memories are small.**
- **Storage hierarchies are large and fast <u>on average.</u>**
- **Parallel processors, collectively, have large, fast memories -- the slow accesses to "remote" data we call "communication".**
- **Algorithm should do most work on local data.**

Load Imbalance

- Load imbalance is the time that some processors in the system are idle due to
  - insufficient parallelism (during that phase).
  - unequal size tasks.
- Examples of the latter
  - adapting to “interesting parts of a domain”.
  - tree-structured computations .
  - fundamentally unstructured problems.
- Algorithm needs to balance load

Parallel Programming for Performance is Challenging

Amber (chemical modeling)



- Speedup(P) = Time(1) / Time(P)
- Applications have “learning curves”

Course Organization

Schedule of Topics

- Introduction
- Parallel Programming Models and Machines
  - Shared Memory and Multithreading
  - Distributed Memory and Message Passing
  - Data parallelism
- Sources of Parallelism in Simulation
- Algorithms and Software Tools (depends on student interest)
  - Dense Linear Algebra
  - Partial Differential Equations (PDEs)
  - Particle methods
  - Load balancing, synchronization techniques
  - Sparse matrices
  - Visualization (field trip to NERSC)
  - Sorting and data management
  - Grid computing
- Applications (including guest lectures)
- Project Reports

Reading Materials

- Three on-line texts:
  - Demmel's notes from CS267 Spring 1999 (mostly similar to 2000 notes).
  - Culler and Singh's book "Parallel Computer Architecture" (CS258 text, first chapter on-line).
  - Ian Foster's book, "Designing and Building Parallel Programming".
- Some papers and books will be placed on reserve.
- The web: www.nersc.gov/~dhbailey/cs267

Computing Resources

- NOW: ~100 Sun Ultrasparcs with a fast network.
- Four clustered Sun Enterprise 5000 8-proc SMPs.
- Millennium prototype: clustered Intel SMPs.
- Assorted other SMPs from IBM, DEC.
- Cray T3E (640 CPUs) at LBL/NERSC.
- Possibly a 16-proc SMP associated with KDI project.

Requirements

- Fill out on-line account registration.
- Fill out on-line survey, including available times for discussion section
- Weekly reading: be ready to discuss in class (10%).
- Four programming assignments (25%).
  - Hands-on experience, interdisciplinary teams.
  - If you don't do it yourself, you'll drop when the project gets interesting.
- Midterm? (20%).
- Final Project (45%).
  - Teams of three - interdisciplinary is best.
  - Interesting applications or advance of systems.

Projects

- Challenging team programming effort on a problem worth solving.
- Conference quality publication.
- Required presentation at end of semester.
- Interdisciplinary (usually).

What you should get out of the course

In depth understanding of:

- **How to apply parallel computers to demanding problems.**
- **Understanding of requirements of parallel applications (and their programmers).**
- **Knowledge of hardware, software, theory and practice of parallel computing.**

First Assignment

- See home page for details.
- Find an application of parallel computing and build a web page describing it.
  - Choose something from your research area.
  - Or from the web or elsewhere.
- Evaluate the project. Was parallelism successful?
- Due one week from today (1/26).